Capitolo 2

Instruction Set

- Il repertorio di istruzioni di un computer
- Computer diversi hanno instruction set diversi
  - Ma con molti aspetti in comune
- I primi computer avevano instruction set molto semplici
  - Per semplificare l'implementazione
- Anche molti computer moderni hanno instruction set semplici

Instruction Set del MIPS

- Usato come esempio per tutto il corso
- Progettato all'Università di Stanford e commercializzato da MIPS Technologies (www.mips.com)
- Grande fetta del mercato dei processori embedded
  - Applicazioni all'elettronica di consumo, archiviazione/rete, dispostivi, videocamere, stampanti, ...
- Rappresentativo di molte ISA moderne
  - Vedi *Scheda tecnica riassuntiva del MIPS* nel libro

Operazioni Aritmetiche

- Addizione e sottrazione, tre operandi
  - Due sorgenti e una destinazione

```
add a, b, c      # a diventa b + c
```

- Tutte le operazioni aritmetiche hanno questa forma
- **Principio di Progettazione 1**: *"La semplicità favorisce la regolarità."*
  - La regolarità rende l'implementazione più semplice
  - La semplicità permette maggiori prestazioni a costi inferiori

Esempio

- Codice C:

```
f = (g + h) - (i + j);
```

- Codice MIPS compilato:

```
add t0, g, h     # temp t0 = g + h
add t1, i, j     # temp t1 = i + j
sub f, t0, t1    # f = t0 - t1
```

Operandi su registro

- Le istruzioni aritmetiche usano operandi **registro**
- Il MIPS usa un **register file** di 32 registri a 32 bit
  - Usati per dati acceduti frequentemente
  - Numerati da 0 a 31
  - Un dato su 32 bit è detto "*word*"
- Nomi assembler:
  - `$t0`, `$t1`, ..., `$t9` per valori temporanei
  - `$s0`, `$s1`, ..., `$s7` per valori salvati
- **Principio di Progettazione 2**: *"Più piccolo è più veloce."*
  - Si pensi alla memoria: milioni di locazioni

Esempio

- Codice C:

```
f = (g + h) - (i + j);
```

  - `f`, ..., `j` in `$s0`, ..., `$s4`

- Codice MIPS compilato:

```
add $t0, $s1, $s2
add $t1, $s3, $s4
sub $s0, $t0, $t1
```